

LOG4J - LOG FORMATTING

Apache log4j provides various Layout objects, each of which can format logging data according to various layouts. It is also possible to create a Layout object that formats logging data in an application-specific way.

All Layout objects receive a `LoggingEvent` object from the Appender objects. The Layout objects then retrieve the message argument from the `LoggingEvent` and apply the appropriate `ObjectRenderer` to obtain the String representation of the message.

The Layout Types:

The top-level class in the hierarchy is the abstract class `org.apache.log4j.Layout`. This is the base class for all other Layout classes in the log4j API.

The Layout class is defined as abstract within an application, we never use this class directly; instead, we work with its subclasses which are as follows:

- `DateLayout`
- [HTMLLayout](#) (Explained in this tutorial)
- [PatternLayout](#). (Explained in this tutorial)
- `SimpleLayout`
- `XMLLayout`

The Layout Methods:

This class provides a skeleton implementation of all the common operations across all other Layout objects and declares two abstract methods.

S.N.	Method & Description
1	public abstract boolean ignoresThrowable() This method indicates whether the logging information handles any <code>java.lang.Throwable</code> object passed to it as a part of the logging event. If the Layout object handles the <code>Throwable</code> object, then the Layout object does not ignore it, and returns false.
2	public abstract String format(LoggingEvent event) Individual layout subclasses will implement this method for layout specific formatting.

Apart from these abstract methods, the Layout class provides concrete implementation for the methods listed below:

S.N.	Method & Description
1	public String getContentType() Returns the content type used by the Layout objects. The base class returns <code>text/plain</code> as the default content type.

2	public String getFooter() Specifies the footer information of the logging message.
3	public String getHeader() Specifies the header information of the logging message.

Each subclass can return class-specific information by overriding the concrete implementation of these methods.